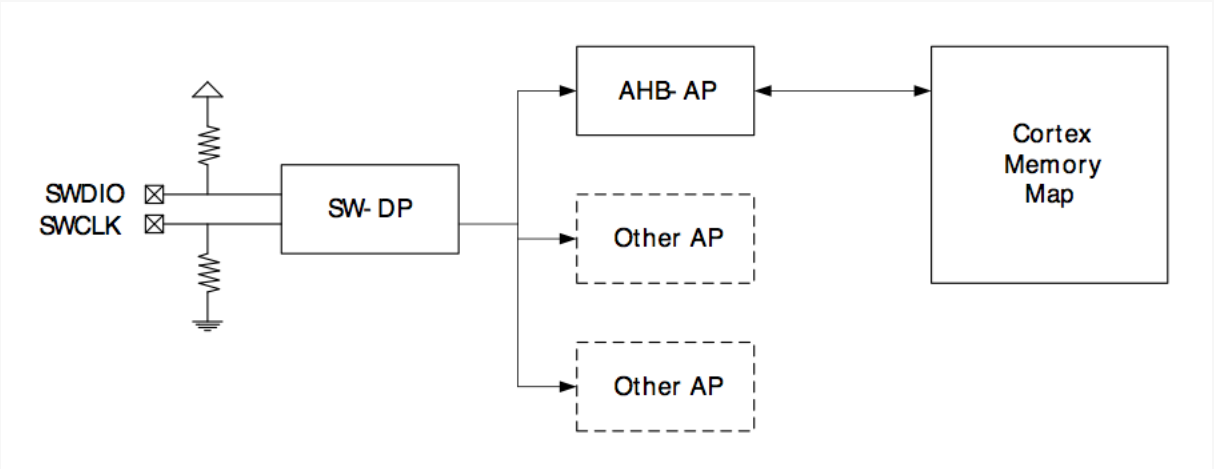
EFM32 Series 0: Debugging and Programming

10 DECEMBER 2013

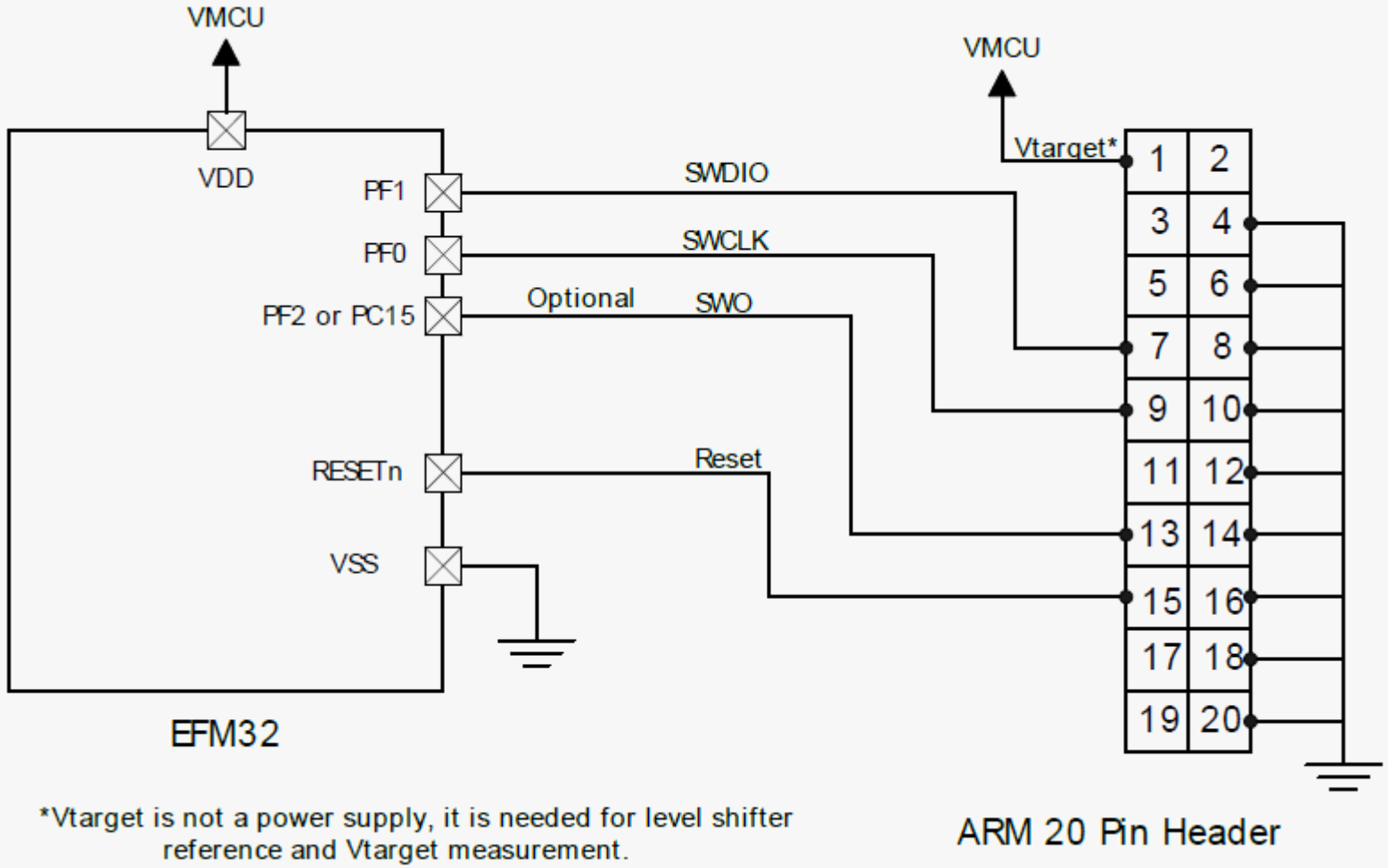
EFM32 Debug Interface

SWD = ARM Serial Wire Debug Interface

- Only two pins required:
  SWDIO + SWCLK
- Optional pin:
  SWO
- No JTAG
- No Boundary Scan
- Connected to core
  through AHB-AP
- Further reading: AN0062



EFM32 Debug Pin-out



*Vtarget is not a power supply, it is needed for level shifter reference and Vtarget measurement.

- SEGGER UM0001 J-Link/J-Trace User Guide:
  - Standard debug connectors

SWD Protocol

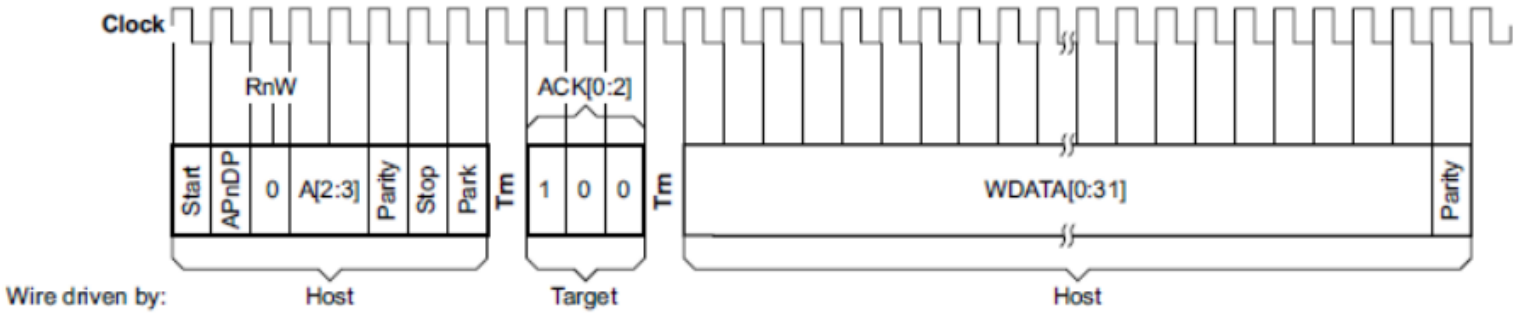
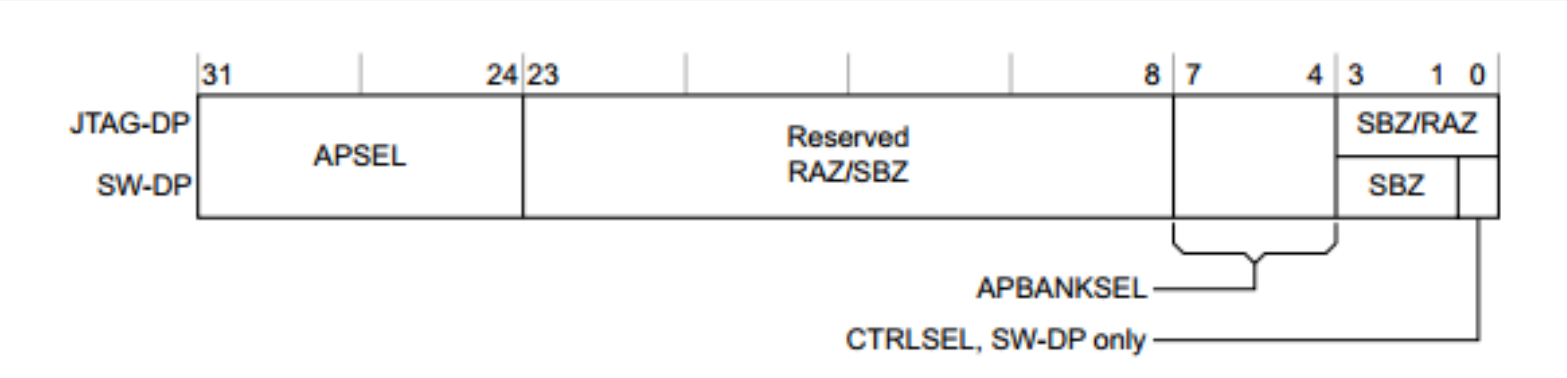


Table 2.1. SW-DP registers

Address	Read	Write
0x00	IDCODE	ABORT
0x04	CTRL/STAT [1]	CTRL/STAT [1]
0x08	RESEND	SELECT
0x0C	RDBUFF	N/A

[1] WCR register if CTRLSEL bit of SELECT is 1, see [adi5]

SWD Select Register



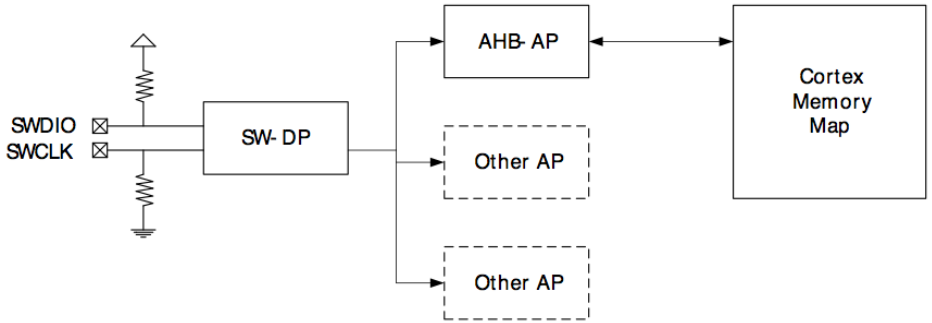
- APSEL selects active AP
  - Only APSEL = 0 is used on EFM32
  - Normally AHB-AP (unless chip is locked)
- APBANKSEL selects current active bank
  - 16 possible banks
- Each AP bank has 4 active registers
  - Total 16 * 4 = 64 registers allowed per AP

AHB-AP

Table 2.2. AHB-AP registers

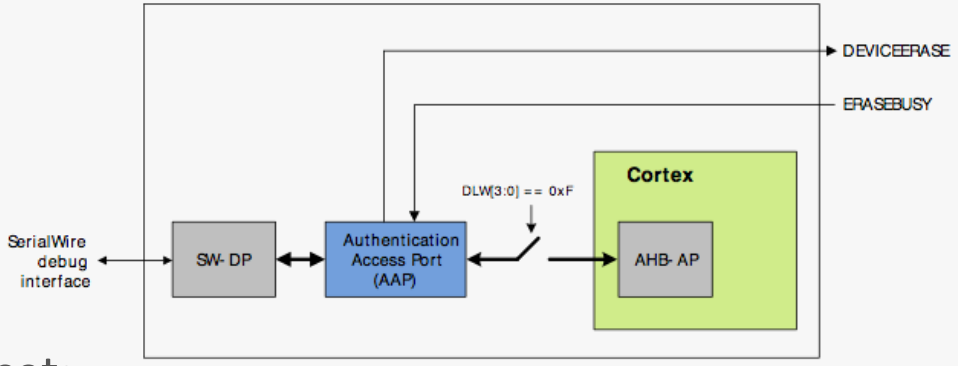
Address	Read	Write
0x00	CSW	CSW
0x04	TAR	TAR
0x08	N/A	N/A
0x0C	DRW	DRW
0xFC	IDR	N/A

- Responsible for accessing internal memory map
- Read operation
  - Write address to TAR
  - Read DRW
- Write operation
  - Write address to TAR
  - Write data to DRW



Debug Lock

- Prevents access to firmware through debug port
- Debug Lock = Disable connection between Debug Port and core



- When device comes out of reset:
  - 1. SWD-DP connected to AAP
  - 2. SWD-DP reads lock word in AAP
  - 3a. If unlocked: AAP opens connection to AHB-AP
  - 3b. If locked: AHB-AP access not open. DP can only access AAP
- AAP – Authentication Access Port
  - Main capability: Mass Erase – erases flash, SRAM and lock bits.
    Mass Erase does NOT erase User Data Page

AAP – Programmers Model

- When ‘Debug Lock’ is enabled, AAP is accessed instead of AHB-AP (on AP #0)
- Debugger can verify Locked status by reading IDR
  - AHB-AP: IDR = 0x24770011
  - AAP: IDR = 0x16E60001

- In J-Link Commander:

```
// First Write 0x000000F0 to SELECT to select
// the last register bank of AP #0.
SWDWriteDP 2 0x000000F0

// Dummy-read the fourth register in this
// bank (A[3:2] == 0b11), this is the IDR register.
SWDReadAP 3

// Read the RDBUFF register to get the
// actual contents of IDR
SWDReadDP 3
```

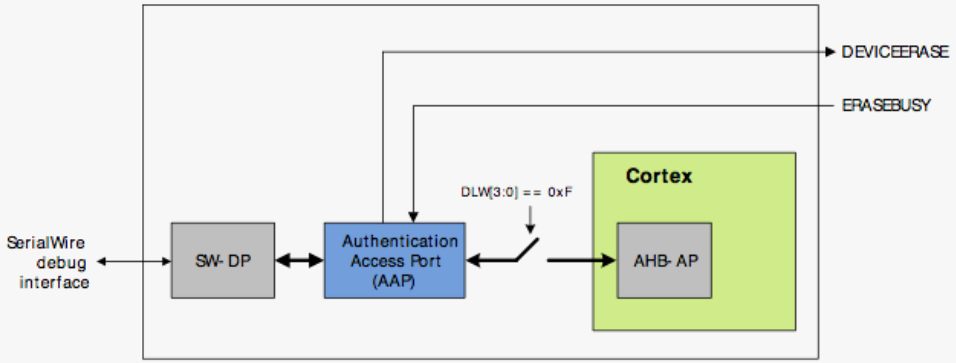
AAP – On M0+ (ZG)

- On ZG AHB-AP is always available and AAP is mapped to internal address 0xF0E00000
- When locked, only AAP is accessible by AHB-AP
- When unlocked AAP is not accessible

```
// First Write 0x00000000 to SELECT to select
// the first register bank of AP #0 (AHB-AP)
SWDWriteDP 2 0x00000000

// Write address of AAP_IDR to the TAR register
SWDWriteAP 1 0xF0E000FC

// Dummy read the DRW register. This will
// generate a memory access to read IDR
SWDReadAP 3

// Read the RDBUFF register to get the
// actual contents of IDR
SWDReadDP 3
```

Set Debug Lock

- Debug Lock is enabled by clearing Debug Lock Word (DLW)
- DLW is part of Lock Bits (LB) Flash page
- Debug Lock is not enforced until after a hard reset
  - Pin reset or power cycle
  - Watchdog reset also works since it reset the debug interface
  - Software reset (NVIC_SystemReset()) is *not enough!*

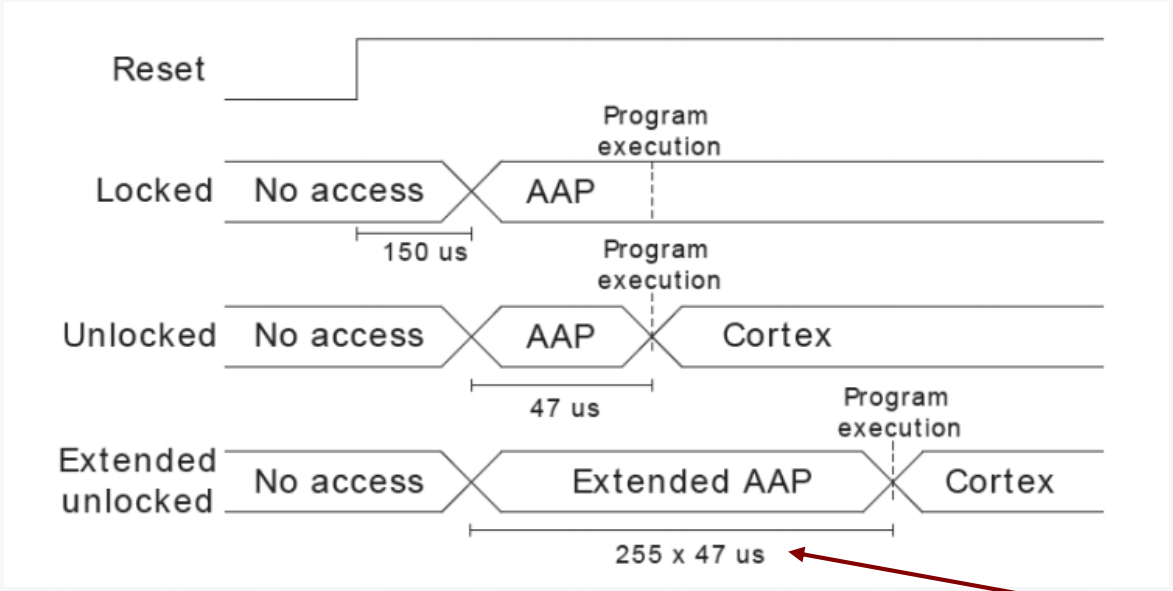


Debug Unlock

Offset	Name	Type	Description
0x000	AAP_CMD	W1	Command Register
0x004	AAP_CMDKEY	W1	Command Key Register
0x008	AAP_STATUS	R	Status Register
0x0FC	AAP_IDR	R	AAP Identification Register

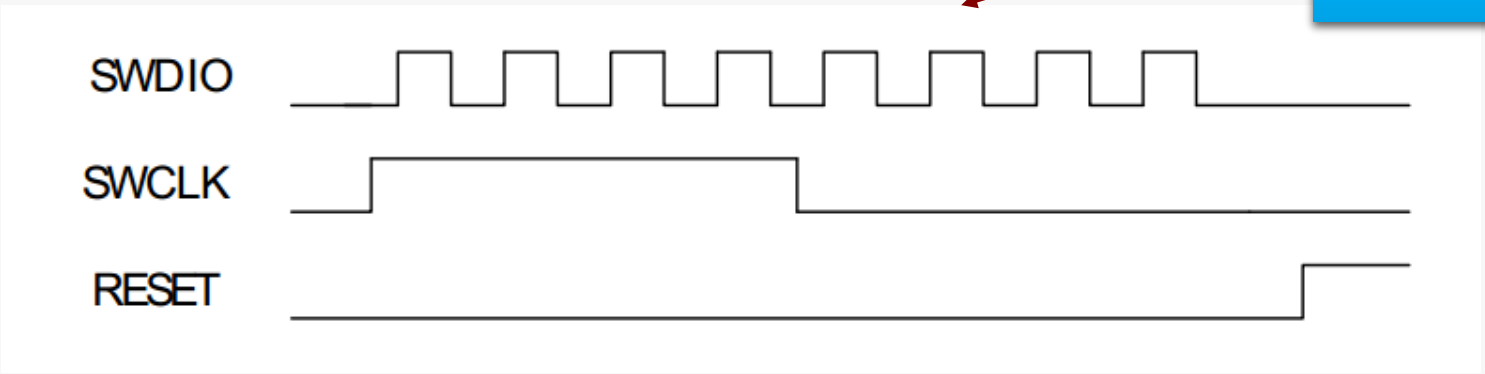
- Unlock sequence
  - Write 0xCFACC118 to AAP_CMDKEY
  - Write 1 to the DEVICEERASE bit of AAP_CMD

AAP Reset Window



Not available on Gecko

Extension sequence



Recover Bricked Device

- How to “brick” an EFM32:
  - Intentionally: Lock debug interface
  - Unintentionally:
    - Disable HF clock
    - Disable debug pins
    - Reconfigure debug pins
    - Enter EM4
  - If this is done early in code, the debugger don’t have time to halt the CPU before the debug interface is disabled!
- Debug Unlock feature
  - Mass erase
  - Remember: User Data Page is not erased
  - Commander unlock sequence is timing critical – be aware of long wires
  - All EFM32 kits can unlock any EFM32 device.
    - No known 3rd party debuggers implement Debug Unlock
    - Currently ZG can not be unlocked by any EFM32 kit. Use AN0062 instead. Will be fixed by firmware update soon.

Debug Modules

- FPB - Flash Patch and Breakpoint unit
  - Breakpoints and code patches
  - 8 HW breakpoints
- DWT - Data Watch point and Trace unit
  - Watchpoints, trigger resources and system profiling
  - 4 configurable comparators: hardware watchpoint, ETM trigger, PC sampler trigger, data address event trigger
  - Counters: clock cycles, folded instructions, LSU, sleep cycles, CPI, interrupt overhead
  - Periodic PC sample output (used in energyAware Profiler)
- ITM – Instrumentation Trace Macrocell (except ZG/M0+)
  - Application-driven trace
  - Trace sources:
    - Software trace
    - Hardware trace
    - Time stamping
  - ITM + SWO = Serial Wire Viewer (SWV)
- ETM – Embedded Trace Macrocell (GG, LG, WG only)
  - Instruction and data trace in real-time
  - 5 extra pins: Trace CLK + Data [3:0]
  - Note – FPGA bug on DK, can’t be used

Factory programmed boot loader

- Factory-programmed boot loader:
  - Two versions of the boot loader exist:
    - UART(non-USB parts)
      - Special version for part with few pins (no UARTn)
    - UART and USB (all USB parts)
- Commands:
  - Upload
  - Destructive upload (overwrites boot loader itself)
  - Write data to User Data page
  - Write Lock Bits page (write/erase protect flash pages)
  - Verify upload and flash contents
  - Boot application
  - Reset device
  - Lock debug interface

How to invoke the USART boot loader

Step-by-step guide for EFM32GG-DK3750:

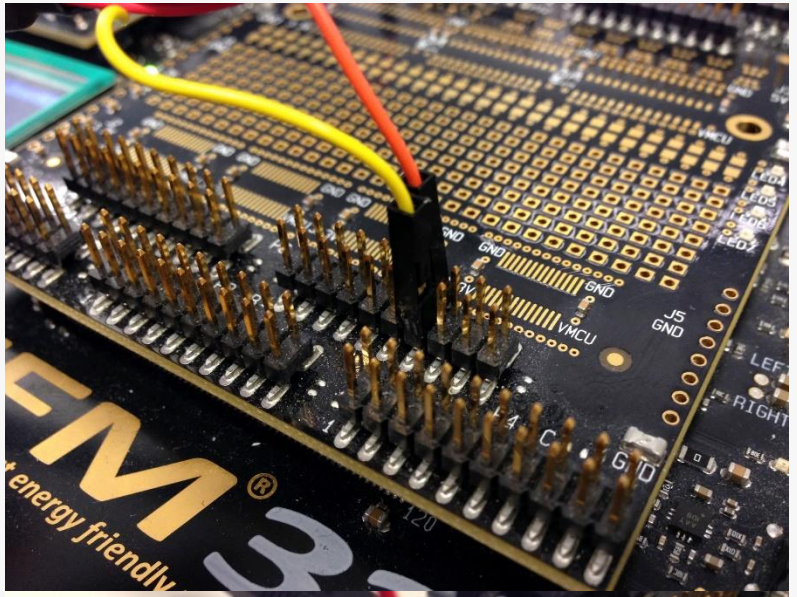
1. Use energyAware Commander to download the boot loader binary from AN0003
2. Connect the USB UART cable to the UART pins on the prototyping board. Connect:
    1. P6.13 (PE10, EFM32 Tx) <--> Yellow (USB Rx)
    2. P6.14 (PE11, EFM32 Rx) <--> Orange (USB Tx)
3. Pull SWCLK high. SWCLK can be found on the prototyping board P8.3 (PF0)
4. While pulling SWCLK high, press the reset button on the MCU board
5. In your terminal emulator, transmit the auto-baud synchronization character 'U' (capital)

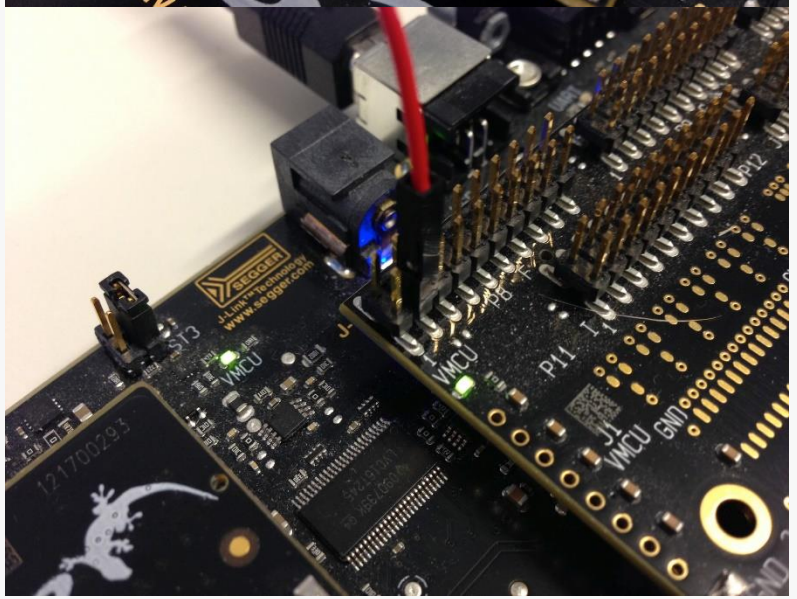
Now, you should get the boot loader prompt in the terminal emulator.

Terminal emulator:

Teraterm

(supports X-MODEM with CRC)



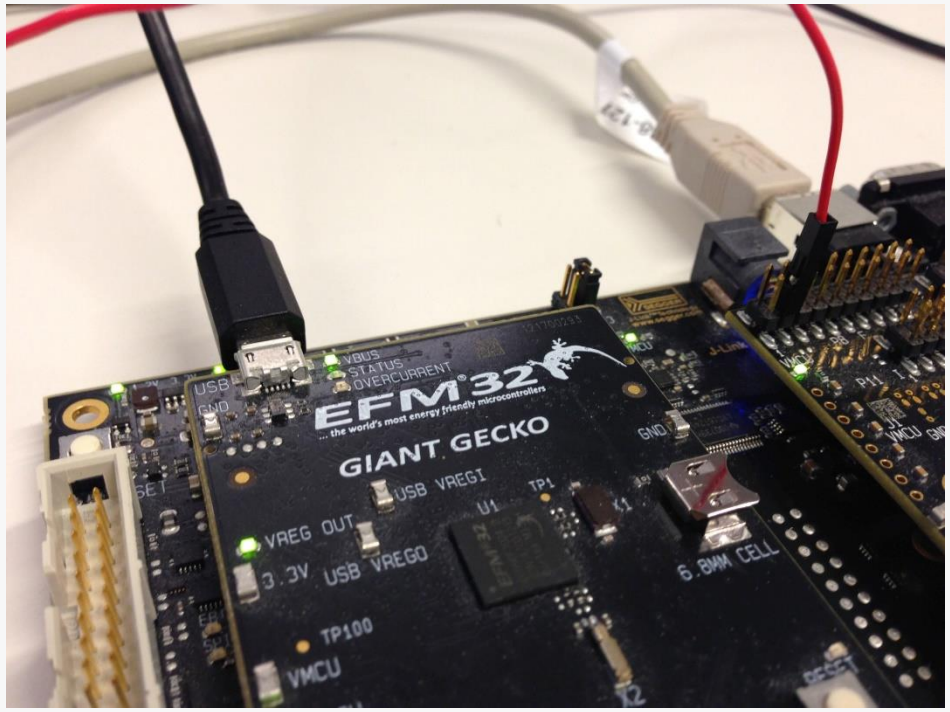


How to invoke the USB boot loader

Step-by-step guide for EFM32GG-DK3750:

1. Use energyAware Commander to download the boot loader binary from AN0042
2. Pull SWCLK high. SWCLK can be found on the prototyping board P8.3 (PF0).
3. While pulling SWCLK high, press the reset button on the MCU board
4. First time only: Install USB CDC virtual UART device driver (EFM32-cdc.inf)
5. Insert micro-USB cable
6. USART mode: Transmit the auto-baud synchronization character 'U' (capital)
   USB mode: Insert USB cable

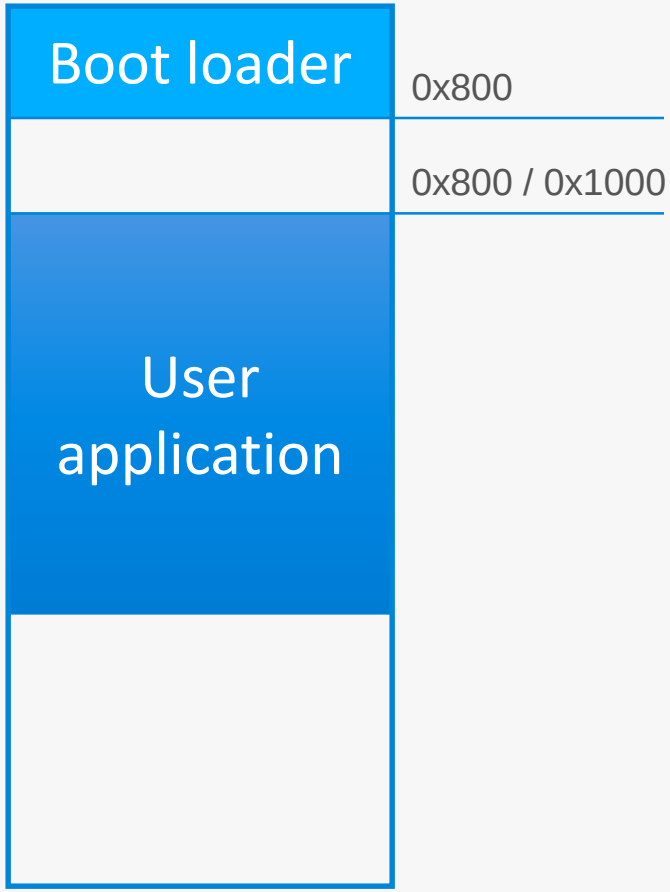
Now, you should get the boot loader prompt in the terminal emulator.



Creating applications for boot loader

- Default: User application on address 0x0
- Destructive upload: no changes required
  (boot loader is overwritten)
- Keep boot loader: User application linked to run from 0x800 / 0x1000 / 0x4000
- Boot loader size:
  - 2 kB: ZG, TG, G
  - 4 kB: LG, GG, WG
  - 16 kB: LG, GG, WG with USB
- Instructions in app. notes
  - IAR: Linker files included in AN
  - Keil MDK-ARM: Change project settings
  - GCC: Edit linkerfile
- IAR debug: Set position of vector table in code:

```
SCB->VTOR = 0x800;
```



Boot Loader Documentation

- Pre-programmed boot loaders documentation:
  - AN0003 UART Boot Loader
  - AN0042 USB-UART Boot Loader
  - Note: Boot loaders difficult to compile
- Boot loader software examples:
  - AN 0060 AES Boot Loader
    - Loads an AES encrypted firmware
    - Backup image – verify new image
  - AN0052 USB MSD Host Boot Loader
    - Loads firmware from USB Mass Storage Device (memory stick)

Debug printf()

- MCU – no standard output for printf()
- Simplicity: Code to retarget printf() output to USART provided
- Easy to use:
  - Add retarget source code to build:
    - *<energymicro>\kits\common\drivers\retargetserial.c*
    - *<energymicro>\kits\common\drivers\retargetio.c*
  - Include
    - *retargetserial.h*
    - *stdio.h*
  - Call *RETARGET_SerialInit()*
  - Use *printf()* to print text to USART
- Note: printf() is a very versatile function – will increase codesize, particularly on GCC.
- GCC: iprintf() (integer support only) reduces codesize

Serial Wire Viewer

- SWV feature in ITM allows character output on SWO pin
- energyAware Commander:
  - Terminal
  - Source code
- ITM_SendChar() = single-character output
- Low energy modes:
  EFM32 can power down the debug interface before the ITM character buffer is empty

SILICON LABS

www.silabs.com/efm32